Simulation-based Queueing Models for Performance Analysis of IoT Applications

Georgios Bouloukakis, Ioannis Moscholios, Nikolaos Georgantas, Valérie Issarny

Simulation-based Queueing Models for Performance Analysis of IoT Applications

Georgios Bouloukakis*‡, Ioannis Moscholios† Nikolaos Georgantas‡ Valérie Issarny‡
*Donald Bren School of Information and Computer Sciences University of California, Irvine, USA.
gboulouk@ics.uci.edu
†Dept. of Informatics & Telecommunications, University of Peloponnese, Tripolis, Greece.
idm@uop.gr
‡MiMove Team, Inria Paris, France.
firstname.lastname@inria.fr

***Abstract*—To facilitate the development of Internet of Things (IoT) applications, numerous middleware protocols and APIs have been introduced. Such applications built atop reliable or unreliable protocols and they expose different characteristics. Additionally, with regard to the application context (e.g., emergency response operations), several Quality of Service (QoS) requirements must be satisfied. To study QoS in IoT applications, the provision of a generic performance analysis methodology is required. Queueing network models offer a simple modeling environment, which can be used to represent IoT interactions by combining multiple queueing model types for building queueing networks. The resulting networks can be used for performance analysis through analytical or simulation models. In this paper, we present several types of queueing models that represent different QoS settings of IoT interactions, such as intermittent mobile connectivity, message drop probabilities, message availability/validity and resource constrained devices. Using MobileJINQS, we simulate our models demonstrating the significant effect on response times and message success rates when varying QoS settings.**

***Index Terms*—Queueing Models, Mobile Connectivity, Internet of Things, QoS Analysis**

I. Introduction

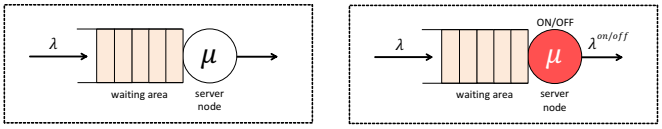
The Internet of Things (IoT) promises the integration of the physical world into computer-based systems. IoT devices, featuring sensing capabilities, are deployed in a variety of application domains, such as smart buildings, community spaces, intelligent transportation to name a few. By exploiting their information, there are new opportunities to improve peoples' safety and quality of life. For instance in [1], [2], IoT devices are used by volunteers to monitor homes/offices for possible indications of seismic activity. Such an application generates critical information and is expected to function correctly, timely and reliably. Hence, identifying a general methodology for the performance modeling and analysis of IoT applications is of prime importance.

IoT devices can be mobile, low-powered, inexpensive and with regard to the environmental context, the resulting applications expose different characteristics. In particular, several Quality of Service (QoS) constraints can be identified: *i)* intermittent connectivity of mobile devices; *ii)* limited memory of inexpensive devices; *iii)* limited availability of data in emergency response scenarios, etc. Furthermore, IoT applications built atop reliable/unreliable protocols and Application Programming Interfaces (APIs) which introduce additional QoS constraints [3], [4]. Hence, these constraints raise important challenges for the supporting of IoT performance analysis.

Existing efforts [5]–[7] concerning the design and evaluation of mobile systems under specific constraints (e.g., intermittent availability, limited resources, etc) and for specific use cases rely on the field of *Queueing Theory* [8]. Key IoT protocols have been evaluated with regard to performance metrics such as response times and delivery success rates [9], [10]. However, such efforts are protocol-specific and thus, the research community has analyzed the performance of well known interactions paradigms [11], [12] that may abstract different IoT protocols. The publish/subscribe paradigm has been analyzed using formal models [13] and evaluated using models such as Queueing Petri Nets [14] or Queueing networks [15].

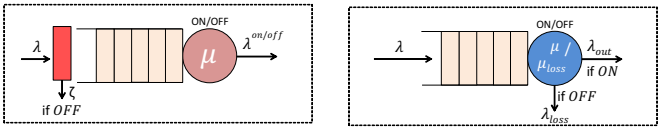
Queueing networks have been extensively applied to represent and analyze communication and computer systems. A *queueing network* is a network of connected service centers which provides analytical or simulation solutions for performance measures (e.g., response time). Combining separate service centers in order to form a queueing network, enable us to model and represent an IoT interaction. Towards this, in [16]–[19] we have identified a methodology where middleware protocol nodes (clients, servers, brokers, etc) are represented as queues and the exchanged messages as jobs served. To model QoS settings such as the intermittent connectivity [20] of mobile Things, we have investigated a separated queueing center via the so called *ON/OFF queueing model* in [16], [17]. Then, we demonstrated the applicability of our approach by modeling the performance of publish/subscribe [16], [18] and data streaming protocols [17]. In [19] we applied our approach to analyze the performance of IoT devices with heterogeneous QoS settings. In this paper, we provide additional queueing models that represent the aforementioned QoS constraints of IoT applications. The key contributions of this work are:

The work is supported by the research associate team ACHOR (inria.fr/en/associate-team/achor).



(a) M/M/1 queue. (b) ON/OFF queue.

Fig. 1: M/M/1 and ON/OFF queues.



(a) Probabilistic ON/OFF queue. (b) ON/OFF loss queue.

Fig. 2: Probabilistic and loss ON/OFF queues.

1) The description of a generic ON/OFF model extended with message join probabilities and message losses.
2) The introduction of additional features for queueing models representing resource constrained devices and message availability.
3) The comparison of different queueing models in terms of response time and message success rate through simulation-based experiments.

The rest of this paper is organized as follows: Section II, provides a description of our queueing models that represent a variety of QoS settings found in IoT applications. In Section III, we compare the performance of queueing models when applying message join probabilities, message availabilities and finite buffer capacities. We finally complement this paper with a brief conclusion and future work in Section IV.

II. Queueing Models

In this section, we define the individual queueing models that are used as part of the simulation-based queueing networks of our methodology.

A. M/M/1 model

This queue models uninterrupted serving (transmission, reception or processing) of messages as part of an end-to-end IoT interaction. It corresponds to the most common M/M/1 queue (see Fig. 1a), featuring Poisson arrivals and exponential service times.

An M/M/1 queue ($q_{\mathrm{m/m/1}}$) is defined by the tuple:

$$q_{\mathrm{m/m/1}} = (\lambda, \mu). \tag{1}$$

where λ is the input rate of messages to the queue and μ is the service rate for the processing of messages. Let D be the service demand for the processing delay of each message (i.e., $D = 1/\mu$). Based on standard solutions for the M/M/1 queue [15], the time that a message remains in the system (corresponding to queueing time + service time; we also call it mean response time) is given by:

$$\Delta^{\mathrm{m/m/1}} = \frac{D}{1 - \lambda D}. \tag{2}$$

B. ON/OFF model

To deal with the mobile peer's connections and disconnections we introduce the *Intermittent (ON/OFF)* queue, which is depicted in Fig. 1b. Messages arrive according to a Poisson process with rate $\lambda > 0$, and are placed in a queue waiting to be "served" (waiting area in Fig. 1b). Messages are served with rate $\mu > 0$, which is exponentially distributed.

We assume that the server is subject to an on-off procedure. That said, it remains in the ON-state for an exponentially distributed time with parameter θ_{ON}, during which it serves messages (if any). Let T_{ON} be the time the server is ON – i.e., $T_{\mathrm{ON}} = 1/\theta_{\mathrm{ON}}$. Upon the expiration of this time the server enters the OFF-state during which it stops working (stops serving relevant messages) for an exponentially distributed time period with rate θ_{OFF}. Let T_{OFF} be the time the server is OFF – i.e., $T_{\mathrm{OFF}} = 1/\theta_{\mathrm{OFF}}$. Accordingly, an ON/OFF queue $q_{\mathrm{on/off}}$ is defined by the tuple:

$$q_{\mathrm{on/off}} = (\lambda, \lambda^{\mathrm{on/off}}, \mu, T_{\mathrm{ON}}, T_{\mathrm{OFF}}). \tag{3}$$

where λ is the input rate of messages to the queue, $\lambda^{\mathrm{on/off}}$ is the output rate of messages, and μ is the service rate for the processing of messages (if any) during T_{ON}. The output process $\lambda^{\mathrm{on/off}}$ is intermittent, because no messages exit the queue during T_{OFF} intervals. Without loss of generality, we make the following assumption: if T_{ON} expires and there is a message currently being served, the server interrupts its processing and will continue in the next T_{ON} period.

Let $\Delta^{on/off}$ be the the mean response time (the time that a message remains in the system) for the $q_{\mathrm{on/off}}$ queue. In our recent works [16], [17], we have investigated an analytical solution for estimating $\Delta^{on/off}$ by considering average periods of connections and disconnections (T_{ON}, T_{OFF}). This is given by:

$$\Delta^{on/off} = \frac{E(n)_{on/off}}{\lambda}. \tag{4}$$

where $E(n)_{on/off}$ is the average number of messages in the system (server + queue). Its formula is defined in [17].

C. Probabilistic ON/OFF model

To reduce queueing delays in the ON/OFF model (see subsection II-B), we introduce the *probabilistic ON/OFF model* which is depicted in Fig. 2a. In such a model, messages arrive in the system according to a Poisson process with rate λ. If the server is online (ON) then messages are placed in a queue waiting to be served. Otherwise, if the server is offline (OFF), messages decide to join the queue with probability ζ or leave the system with probability $1-\zeta$. A probabilistic ON/OFF queue $q^{prob}_{\mathrm{on/off}}$ is defined by the tuple:

$$q^{prob}_{\mathrm{on/off}} = (\lambda, \lambda^{\mathrm{on/off}}, \mu, \zeta, T_{\mathrm{ON}}, T_{\mathrm{OFF}}). \tag{5}$$

where λ is the input rate of messages, ζ is the probability of joining the queue, $\lambda^{\mathrm{on/off}}$ is the output rate of messages and

μ is the service rate for the transmission of messages (if any) during T_{ON}. It is worth noting that if $\zeta = 1$, then all messages join the system during OFF periods. This is equivalent with the ON/OFF model presented in the subsection II-B. Hence, the probabilistic ON/OFF model generalizes the ON/OFF model. $\Delta_{prob}^{on/off}$ is the the mean response time (the time that a message remains in the system) for the $q_{\text{on/off}}^{prob}$ queue. We are currently working on identifying an analytical model for estimating $\Delta_{prob}^{on/off}$.

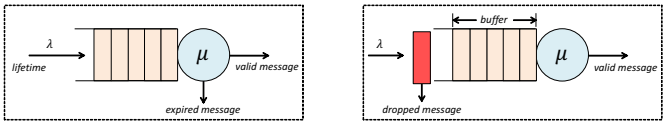
Using the probabilistic ON/OFF model, system designers will be able to appropriate tune/improve delays of components that may disconnect for an average period T_{OFF} and transmit messages during T_{ON} over a limited bandwidth. Reducing the arrival rate of messages during T_{OFF}, decreases queueing delays and prevents network flooding during T_{ON}. Such a component may be a resource-constrained (in terms of energy and memory) IoT sensor that transits messages over limited bandwidth and may be on sleep-mode periodically (ON/OFF). This probabilistic behavior (i.e., the probability of transmitting messages during T_{OFF}) must be defined from the corresponding application-layer of the IoT device.

D. ON/OFF loss model

The ON/OFF models defined in the above subsections do not assume message drops (or losses) from the system – any message is either served or buffered and then served. The probabilistic ON/OFF model assumes messages losses, however such messages do not enter the queue since they decide to join the queue with probability ζ.

To model message losses we introduce the *ON/OFF loss* queue (see Fig. 2b). Similarly to the above ON/OFF queues, the arrival rate (λ) of messages to the system is Poisson. If the server is ON then messages are placed in a queue waiting to be "served" with service rate μ. Otherwise, (i.e., server is OFF) all messages that join the queue (ζ = 1), are "served" with service rate μ_{loss} and afterwards they exit the system (i.e., we have lost messages). We assume that lost messages are "served" with service rate $\mu_{\text{loss}} > \mu$.

The latter assumption is introduced due to the following reason: we have identified the ON/OFF loss queue in order to model the performance of an IoT mobile device that employs a middleware protocol. Such a protocol introduces message losses since it builds atop UDP and it does not setup any logical session between the sender and the receiver IoT devices. Messages are sent without any guarantee to the other endpoint. As already defined, the service rate μ can be parameterized based on the network transmission delay of the corresponding end-to-end interaction (sender to receiver). On the other hand, to parameterize the μ_{loss} we must identify at which point of the network messages are lost. For instance, assuming that the ON/OFF loss queue models a wireless receiver and its intermittent connectivity, then μ_{loss} can be parameterized through the corresponding network transmission delay. This delay can be defined based on the interaction between the sender and the access point that the receiver connects and disconnects.



(a) Applying `lifetime` periods to a queue.

(b) Applying a buffer size (K) to a queue.

Fig. 3: Queues with event expirations and finite capacity.

An ON/OFF loss queue $q_{\text{on/off}}^{loss}$ is defined by the tuple:

$$q_{\text{on/off}}^{loss} = (\lambda, \lambda_{\text{out}}, \lambda_{\text{loss}}, \mu, \mu_{\text{loss}}, T_{\text{ON}}, T_{\text{OFF}}). \quad (6)$$

where λ is the input rate of messages, λ_{out} and μ are the output and processing rates of messages during the ON periods, λ_{loss} and μ_{loss} are the lost and processing rates during the OFF periods. The λ_{out}/λ_{loss} rates are both exponential, since arrived messages split based on the T_{ON}/T_{OFF} intervals. $\Delta_{loss}^{on/off}$ is the the mean response time for the $q_{\text{on/off}}^{loss}$ queue. We are currently working on identifying an analytical model for estimating $\Delta_{loss}^{on/off}$.

E. Additional Features

Up to now, we have defined queueing models having buffers with infinite capacity and arriving messages with infinite `lifetime` (i.e., a time-to-live period which defines the message's availability inside the queueing network). This certainly affects the response time but also the rate of messages successfully served over the total number of arriving messages. However, modeling IoT interactions with such characteristics may not be realistic. For instance, upon a long disconnection period (e.g., 30 mins) of an IoT sensor, the produced data/messages may exceed the sensor's buffer capacity and/or some of the oldest data may become obsolete for the receiving application/user. Accordingly, in this subsection we introduce the corresponding features that take into account the above constraints. These features can be applied to the queueing models defined in the previous subsections.

1) Queueing network with lifetime messages: As already pointed out, a queueing network is a network of connected queues which can be used to model the performance of a system. For instance, in the context of our work, we model IoT interactions by creating queueing networks which consist of queueing models presented in the previous subsections. As depicted in Fig. 3a, messages arrive with a rate λ in order to be processed in the first queue of the queueing network. An arriving message carries a `lifetime` period attributed to it upon its creation, which represents the message validity inside the queueing network. Hence, a message may enter the queueing network and as soon as its `lifetime` elapses, the message leaves the network and is considered as expired.

To consider a message as expired, we take into account the time the message spends in both queue and server at each queue. Assuming that a queueing network consists of a single M/M/1 queue, a message reneges if its service does not begin by a certain `lifetime` period (which includes

its expected service demand as well). Based on the queueing theory literature, such a model is studied as an *M/M/1 queue with reneging* or *impatient customers* [21]–[23]. In this work, we provide the simulation of the above M/M/1 model through our simulator. Furthermore, we enrich the ON/OFF models to support *reneging* or *impatient customers*. Hence, system designers are able to create queueing networks that may include different queueing models (described in the previous subsection) enriched with lifetime periods.

2) Queues with finite capacity: This is a well known queueing model feature, where a specific buffer size is applied to the queue that ensures having max K messages in the system (queue + server). This prevents from storing too many messages for too long in devices with limited hardware capacity (memory, hard disk). In particular, as depicted in Fig. 3b, messages arrive in the queue with λ_{in}. Before a message enters the queue the following condition is checked: $new\_queue\_size + message\_in\_service > K$. If the condition is true, the message is *dropped*. Otherwise, the message enters the queue to be processed.

Based on the literature, an M/M/1 queue with finite capacity is notated as M/M/1/K [8]. In our models, we represent M/M/1 and the ON/OFF queues presented in the previous subsections with finite capacity by adding the system size (K) to the corresponding definition (see tuples 1, 3, 5, 6).

III. Experimental Results

In [17], we presented our simulator – *MobileJINQS*, which is an extension of the Java Implementation of a Network-of-Queues Simulation (JINQS) [24]. In this work we further extend MobileJINQS to implement the ON/OFF models described in Section II.

For our experimental setup, each different ON/OFF model is utilized to represent a mobile sender with different QoS settings when sending messages. Such a mobile peer generates approximately 7,500,000 messages to accurately calculate per-message mean response times and success rates. It connects and disconnects in the scale of seconds. To represent such behavior for any experiment, we set the ON/OFF model parameters as follows: *i)* the server remains in the ON and OFF states for exponentially distributed time periods T_{ON} = T_{OFF} = 20 sec, thus, the server changes its state every 20 sec; *ii)* messages are processed with a mean service demand D = 0.125 sec and *iii)* messages arrive to the queue with a mean arrival rate λ, varying from 0.05 to 3.9 messages per sec (λ_{max} = 3.9 messages/sec). Then, we setup 4 different experiments which correspond to different ON/OFF models by applying the following QoS settings: **1)** join probability (ζ) = 1, message availability (`lifetime`) = ∞ and buffer capacity (K) = ∞; **2)** ζ = 1, `lifetime` = 30 sec, K = ∞; **3)** ζ = 0.75, `lifetime` = ∞, K = ∞; and **4)** ζ = 1, `lifetime` = ∞, K = 100.

A. Response Times

For each one of the above (4) experiments we run its simulator model (based on the corresponding ON/OFF model) 10 times and average across these runs to derive the simulated curves depicted in Fig. 4. As expected, the highest level of mean response times (∼10-360 sec) are noticed in the 1st experiment where all messages must be buffered and served (without any message losses). At any λ rate, the lowest level of mean response times (∼5-10 sec) are noticed in the 4th experiment when applying lifetime periods at every message. When applying a deterministic buffer capacity (3rd experiment), we notice mean response times from 10 to 16 sec. Finally, the 4th experiment, provides low mean response times (∼8-14 sec) for λ = 0.05 - 2 and higher response times (∼17-49 sec) for λ = 2.5 - 3.9.

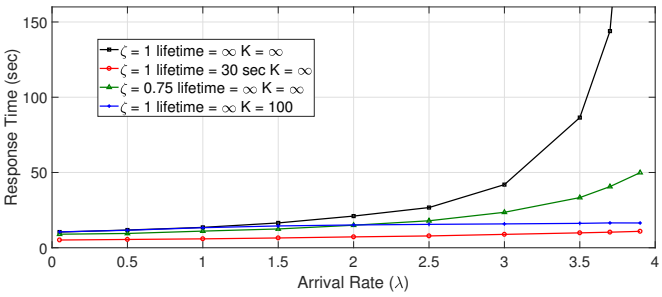


Fig. 4: Response Time of different ON/OFF models.

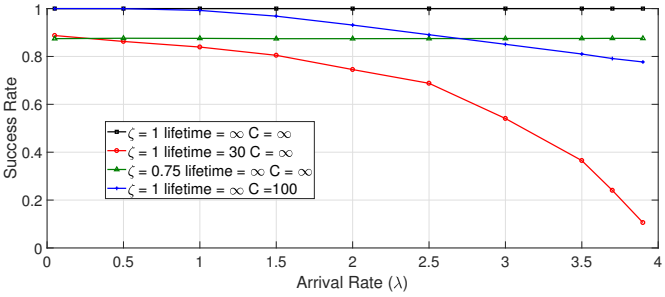


Fig. 5: Success Rates of different ON/OFF models.

B. Success Rate vs. Response Time

In order to study the trade-off between response times and success rates, we present the rates of successful message service in Fig. 5 for the same set of experiments. As expected, there is a 100% of message success rate in the 1st experiment since there are not message losses. On the other hand, the lowest rates are noticed when applying lifetime periods, especially when λ rates increase. In the 3rd experiment, low λ rates provide high message success rates (∼90%-100%) and higher λ rates decrease them (∼87%-77%). Finally, a stable message success rate (75%) can be achieved through the 4th experiment.

IV. Conclusion

Modeling the performance of IoT applications is a tedious task. Messaging in IoT may result to high message delays or message losses due to several QoS settings, such as message availability, intermittent connectivity, resource constrained devices, etc. In this paper, we define simulation queueing models

that represent such QoS settings. Using the simulator we have developed, we evaluate the trade-off between response times and message success rates for different queueing models. By relying on our models, system designers are able to create queueing networks representing IoT interactions and analyze their performance or even tune it based on our experimental results.

In our future work, we intend to identify analytical models for queueing networks with lifetime periods and finite capacity buffers, as well as for the ON/OFF probability and loss models. Then, we plan to model the performance of IoT applications for domains such as emergency response scenarios.

REFERENCES

[1] "Community Seismic Network." http://www.communityseismicnetwork.org, May 2015.

[2] E. Cochran, J. Lawrence, C. Christensen, and A. Chung, "A novel strong-motion seismic network for community participation in earthquake monitoring," *IEEE Instrumentation & Measurement Magazine*, vol. 12, no. 6, pp. 8–15, Jan. 2009.

[3] N. De Caro, W. Colitti, K. Steenhaut, G. Mangino, and G. Reali, "Comparison of two lightweight protocols for smartphone-based sensing," in *IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT)*, Namur, Belgium, Nov. 2013.

[4] S. Lee, H. Kim, D.-k. Hong, and H. Ju, "Correlation analysis of mqtt loss and delay according to qos level," in *International Conference on Information Networking (ICOIN)*, Bangkok, Thailand, Jan. 2013.

[5] F. Mehmeti and T. Spyropoulos, "Performance analysis of "on-the-spot" mobile data offloading," in *Global Communications Conference (GLOBECOM), 2013 IEEE*, Atlanta, GA, USA, Dec. 2013.

[6] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong, "Mobile data offloading: How much can wifi deliver?" in *Proceedings of the 6th International COnference*, Philadelphia, PA, USA, Nov. 2010.

[7] H. Wu and K. Wolter, "Tradeoff analysis for mobile cloud offloading based on an additive energy-performance metric," in *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools*, Bratislava, Slovakia, Dec. 2014.

[8] D. Gross, J. Shortle, J. Thompson, and C. Harris, *Fundamentals of queueing theory*. John Wiley & Sons, 2008.

[9] L. Durkop, B. Czybik, and J. Jasperneite, "Performance evaluation of m2m protocols over cellular networks in a lab environment," in *18th International Conference on Intelligence in Next Generation Networks (ICIN)*, Paris, France, Feb. 2015.

[10] K. Fysarakis, I. Askoxylakis, O. Soultatos, I. Papaefstathiou, C. Manifavas, and V. Katos, "Which iot protocol? comparing standardized approaches over a common m2m application," in *IEEE Global Communications Conference (GLOBECOM)*, Washington DC, USA, Dec. 2016.

[11] L. Aldred, W. M. van der Aalst, M. Dumas, and A. H. ter Hofstede, "On the notion of coupling in communication middleware," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, Agia Napa, Cyprus, Nov. 2005.

[12] A. Kattepur, N. Georgantas, G. Bouloukakis, and V. Issarny, "Analysis of timing constraints in heterogeneous middleware interactions," in *International Conference on Service-Oriented Computing*, Goa, India, Nov. 2015.

[13] F. He, L. Baresi, C. Ghezzi, and P. Spoletini, "Formal analysis of publish-subscribe systems by probabilistic timed automata," in *International Conference on Formal Techniques for Networked and Distributed Systems*, Tallinn, Estonia, June 2007.

[14] S. Kounev, K. Sachs, J. Bacon, and A. Buchmann, "A methodology for performance modeling of distributed event-based systems," in *11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, Orlando, FL, USA, May 2008.

[15] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative system performance: computer system analysis using queueing network models*. Prentice-Hall, Inc., 1984.

[16] G. Bouloukakis, N. Georgantas, A. Kattepur, and V. Issarny, "Timeliness evaluation of intermittent mobile connectivity over pub/sub systems," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, L'Aquila, Italy, Apr. 2017.

[17] G. Bouloukakis, I. Moscholios, N. Georgantas, and V. Issarny, "Performance modeling of the middleware overlay infrastructure of mobile things," in *IEEE International Conference on Communications*, Paris, France, May 2017.

[18] G. Bouloukakis, R. Agarwal, N. Georgantas, A. Pathak, and V. Issarny, "Leveraging cdr datasets for context-rich performance modeling of large-scale mobile pub/sub systems," in *IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Abu Dhabi, UAE, Oct. 2015.

[19] G. Bouloukakis, "Enabling emergent mobile systems in the iot: from middleware-layer communication interoperability to associated qos analysis," Ph.D. dissertation, Inria Paris, Aug. 2017.

[20] G. Bajaj, G. Bouloukakis, A. Pathak, P. Singh, N. Georgantas, and V. Issarny, "Toward enabling convenient urban transit through mobile crowdsensing," in *IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*, Las Palmas, Canary Islands, Spain, Sep. 2015.

[21] A. Montazer-Haghighi, J. Medhi, and S. G. Mohanty, "On a multiserver markovian queueing system with balking and reneging," *Computers & Operations Research*, vol. 13, no. 4, pp. 421–425, 1986.

[22] M. Abou-El-Ata and A. Hariri, "The M/M/c/N queue with balking and reneging," *Computers & Operations Research*, vol. 19, no. 8, pp. 713–716, 1992.

[23] D. Yue, Y. Zhang, and W. Yue, "Optimal performance analysis of an M/M/1/N queue system with balking, reneging and server vacation," *International Journal of Pure and Applied Mathematics*, vol. 28, no. 1, pp. 101–115, 2006.

[24] T. Field, "Jinqs: An extensible library for simulating multiclass queueing networks, v1. 0 user guide," Aug. 2006.